DS Project 2

0416303 楊博凱 資工資電

這次作業是要我們建立一個 Threaded Binary Search Tree，跟以往自己實作 Threaded Binary Tree 與 Binary Search Tree 大不相同。在我建立 Threaded Binary Search Tree 中，head->right 會指向第一個 node，但是 node->left 是指向 NULL，而非 head；反之，tail->left 會指向最後一個 node 但是 node->right 也是指向 NULL，而非 tail。

在 insertion 函式中，我一開始就先讓 num (caculate how many nodes in the totum) 加 1，如果 num==1，代表要 insert 第一個 node 即為 root。若 num!=1，代表 Tree 中已經有 node 存在，因此開始比較 input s 與 now->number 的大小，若 s < now->number 則 go left，反之則 go right 直到找到 leaf (即 now->is_threadr==0&&now->right!=NULL 或是 now->is_threadl==0&&now->left!=NULL)，則 insert 此新的 node。由於經過 insert 後 head 可能指向的已不是第一個 node，tail 指向的也可能不是最後一個 node，因此要 updata head 以及 update tail。

Deletion 函式是本次作業中我覺得最有技巧性的 function，因為除了刪除 node 之外，還要指定 thread 指向的元素；一開始我先用 now 與 pre 兩指標尋找要刪除的元素 s，若 now->number==s 則要刪除的 node 存在，反之 s 不存在於 Tree 中。如果 now->number==s 則先將 num-1，並判斷 now==root 或 now!=root，因為他關係到指標 pre 有無意義。如果 now==root，則判斷 root 有 0, 1 或 2 個 child，若 no child 則讓 root=0。若 root 只有 left child 則讓 root->number 等於 left child 中最大的 node->number 並刪除此 node；若 root 只有 right child 則讓 root->number 等於 right child 中最小的 node->number 並刪除此 node；若 root 有 two childern 則抓取 left subtree 中最大的 node 作為 root，並刪除該元素。若 now!=root，則一

樣判斷 now 有 0, 1 或 2 個 child；若 no child 則直接刪除，只有 left child 則判斷 now=pre->left 或是 pre->right 並讓其指向 now-> left，只有 right child 則判斷 now=pre->left 或是 pre->right 並讓其指向 now->right。若有 two children 則一樣判斷 now=pre->left 或是 now=pre->right；若 now=pre->left，則讓 now->number 等於 left subtree 中最大的 node 之 number，並刪除該 node；若 now=pre->right，則讓 now->number 等於 right subtree 中最小的 node 之 number，並刪除該 node。最後，由於經過 deletion 後的 head 可能指向的已不是第一個 node，tail 指向的也可能不是最後一個 node，因此要 updata head 以及 update tail。

因為這是一個 Threaded Binary Search Tree 因此透過 head 及 tail 我們可以得到 node 的 in-order 排列，即為依照數字大小排序後的結果。因此 inorder_run( )會從 head 指標所指向的第一個 node 開始 print 直到 tail 則終止；而 reverseorder_run( )會從 tail 指標所指向的第一個 node 開始 print 直到 head 則終止。

```
D:\Data Structure\HW2>g++ Source.cpp -o Source.exe

D:\Data Structure\HW2>Source.exe test1.txt
Change! Change myself into a cute mahou shoujo!!
The path: 2 3 5 6

Back! Back to the original life!!
The reverse path: 6 5 3 2


D:\Data Structure\HW2>Source.exe test2.txt
Change! Change myself into a cute mahou shoujo!!
The path: 1 2 3 4 5 6 7 8 9 10 11 12

Back! Back to the original life!!
The reverse path: 12


D:\Data Structure\HW2>Source.exe test3.txt
Change! Change myself into a cute mahou shoujo!!
The path: 1 2 3 4 5 6 7 8 9 10 11 12

Back! Back to the original life!!
The reverse path: 1


D:\Data Structure\HW2>Source.exe test4.txt
Change! Change myself into a cute mahou shoujo!!
The path: 14 16 23 24 29

Back! Back to the original life!!
The reverse path: 29 24 23 16 14


D:\Data Structure\HW2>
```

((運用並命令提示字元 compile 後的結果